Non-restricted Access to Model Solutions : A Good Idea?

Nygren, Henrik

ACM
2019-07-02

Nygren , H , Leinonen , J & Hellas , A 2019 , Non-restricted Access to Model Solutions : A Good Idea? in Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education . ACM , New York, NY , pp. 44-50 , ACM Conference on Innovation and Technology in Computer Science Education , Aberdeen , United Kingdom , 15/07/2019 . https://doi.org/10.1145/3304221.3319768

http://hdl.handle.net/10138/315986
https://doi.org/10.1145/3304221.3319768

unspecified
acceptedVersion

Non-restricted Access to Model Solutions: A Good Idea?

Henrik Nygren
University of Helsinki
henrik.nygren@helsinki.fi

Juho Leinonen
University of Helsinki
juho.leinonen@helsinki.fi

Arto Hellas
University of Helsinki
arto.hellas@cs.helsinki.fi

ABSTRACT

In this article, we report an experiment where students in an introductory programming course were given the opportunity to view model solutions to programming assignments whenever they wished, without the need to complete the assignments beforehand or to wait for the deadline to pass. Our experiment was motivated by the observation that some students may spend hours stuck with an assignment, leading to non-productive study time. At the same time, we considered the possibility of students using the sample solutions as worked examples, which could help students to improve the design of their own programs. Our experiment suggests that many of the students use the model solutions sensibly, indicating that they can control their own work. At the same time, a minority of students used the model solutions as a way to proceed in the course, leading to poor exam performance.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

model solutions, introductory programming, sample solutions, self-regulation

ACM Reference Format:
Henrik Nygren, Juho Leinonen, and Arto Hellas. 2019. Non-restricted Access to Model Solutions: A Good Idea?. In *Innovation and Technology in Computer Science Education (ITiCSE '19), July 15–17, 2019, Aberdeen, Scotland UK*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3304221.3319768

1 INTRODUCTION

Students come to introductory programming courses with a wide variety of backgrounds. Some have programmed before, some have not, some have studied for years, and some are just starting their studies. In the same way as students' backgrounds differ, so do their approaches to studying. Some are driven to deeply understand what they are learning, while some put in the minimum effort needed to complete course work [17, 18]. Some may even avoid learning altogether through piggybacking or using solutions from others [1, 13, 22].

The difference between solutions from others and examples in course materials, in the end, is not vast – what matters is what students do with the content. Examples in course materials, demonstrations, and worked examples can all be beneficial for learning [8, 23]. Regardless of the source of the examples, for learning to be successful, the student is expected to work on their own solutions, often following the principles set out by the examples and the teachers [7]. We are interested in seeing what would happen if the use of "solutions from others" – here, model solutions to programming problems given by us – would be legitimized, and students could use the model solutions to support their learning.

Such knowledge is particularly interesting in programming, where learning – like when learning any complex cognitive skill – takes plenty of effort and practice [20]. One of the particular characteristics in programming is that programming languages are typically highly vulnerable to even small mistakes, and a simple typo can take a significant amount of time to fix [2, 9]. Furthermore, a single character may change the functionality of a program completely, or even stop the program from working at all.

In this article, we discuss an experiment where we changed the dynamics of our introductory programming course and allowed our students to freely access model solutions. Students could view the model solution of each course assignment without the need to complete the assignment or to wait until the deadline of the assignment passed. This closely resembles the approach taken in e.g. many primary and secondary education mathematics books, where students have access to solutions so that they can verify their work. In our case, however, as we are working with computer programs, students have the access to full source code which corresponds to one of the possible ways to solve each particular assignment.

Perhaps the closest match to our experiment is the work on intelligent tutoring systems [4], some of which allow students to ask for hints as they are working on problems. A tutoring system may suggest the next meaningful step, provide guidelines on how to proceed, or show source code to the student [3, 5, 19]. Intelligent tutoring systems do, however, often base showing hints or guiding the students on students' previous behavior, while in our experiment, students' behavior does not change whether they have access to the solutions or not. Moreover, intelligent tutoring systems often have a vast repository of problems that the student can work on.

This article is organized as follows. Next, in Section 2, we outline related research on model solutions, focusing on the use of examples in instruction. Then, in Section 3, we outline the overall design of our experiment, including the study context, data, and the research questions. This is followed by the methodology and the results of our experiment in Section 4. In Section 5, we discuss the results and implications of our experiment, and then in Section 6, we conclude the article and point possible future directions for the work.

ITiCSE '19, July 15–17, 2019, Aberdeen, Scotland UK

ACM ISBN 978-1-4503-6301-3/19/07…$15.00
https://doi.org/10.1145/3304221.3319768

2 BACKGROUND

A model solution, by definition, is a solution to a problem that a student can use to compare their own solution with. Model solutions are typically made available after the student has solved the problem or worked on the problem for a sufficient time, for example until the deadline of the problem has passed. The extent to which the model solution captures the process by which the outcome was achieved varies greatly. For example, some mathematics books outline only expected numeric outcomes, while in programming, model solutions may contain the whole expected source code.

Model solutions are beneficial for verifying results and the thought process and they can be used to increase reflection. Students may, for example, compare and contrast their own solution and the solution given by a teacher. Model solutions are different from students' own previous work in that, while students' own previous work can be similar to model solutions, students have constructed the solutions in their own work either by themselves, in collaboration with others, or by following an example. It is known that having a repository of previous self-constructed solutions can be beneficial [11], as they can then be referred to when facing new problems.

Model solutions can be distinguished from examples. Examples do not typically contain immediate answers to the problems that the students are working on but provide similar content that the student is expected to internalize and then use to solve the given problems. One should take care, however, as too challenging learning materials and examples may hinder students' progress [10], and the quality of the examples influences the quality of students' solutions [12]. Two particular types of examples used to support learning are worked examples [23], which are step-by-step examples showing how a problem should be solved, and modeling [6, 7], which means the process of a teacher showing how a particular task is accomplished. The effectiveness of these approaches is influenced by students' attention, effort, and previous knowledge – for example, guidance that is beneficial for novices may lead to negative outcomes if targeted to more advanced students [14].

Examples that students are given access to should be relevant to the task they are working on. Studying an example and completing an assignment related to the example can lead to better performance, and working on additional assignments helps learning over viewing examples [24]. Even if students are given examples of work that they are expected to do, however, students' differences influence how they work with the examples. For example, poorly performing students may copy content verbatim, while advanced students may try to solve the subsequent problems themselves but only refer to examples when stuck, wanting to check a step, or wanting to avoid a more complex task such as detailed calculation [25].

3 EXPERIMENT DESIGN

3.1 Context and data

Our experiment was conducted in a seven-week introductory programming course offered at the University of Helsinki. Students in the course learn the principles of procedural and object-oriented programming using Java. The course starts with basic input and output, conditional statements and loops, and continues with lists. This is followed by the construction of objects that are mostly used to represent data. Finally, students learn to use maps and learn about basic algorithms that are used to search and sort data. The course has a total of seven programming assignment sets with a total of 177 programming assignments. Each assignment set corresponds to a week in the course and has a deadline. At the end of the course, students attend an exam. Each assignment has a corresponding model solution that includes source code of a working solution to the assignment. The course uses an online course material with embedded videos, questionnaires, and programming assignment handouts. The programming assignments are worked on in a separate programming environment, from where they are submitted to an automated assessment system that grades the work and provides feedback to the students (see [28]). There are walk-in labs that students can attend simply to work on their own or to ask questions from course assistants and course teachers – the labs are available for at least 20 hours each week (for more details on the course organization, see e.g. [26, 27]).

In the previous course iterations, students have been able to view model solutions to programming assignments once they have either completed the programming assignment or once the assignment deadline has passed. In our experiment, we removed the restrictions that limited access to the programming assignment model solutions. Instead of students having to complete the assignment or to wait until the deadline of the assignment set passed, students could access the model solutions whenever they wished, even before starting to work on the actual assignment. Direct copy-pasting of solutions was strongly discouraged, and lectures also discussed the effort that learning to program requires.

In order to view the model solutions, students had to log in to the automated assessment system. Every time that a student viewed a model solution, the system stored the student's id, the programming assignment that was viewed, and the timestamp. Similarly, when students submit their work for grading, the automated assessment system stores students' information. In addition to the logs from the automated assessment system, we have access to the results of the course, and have conducted semi-formal interviews with 16 students and teaching assistants on the course.

When comparing this iteration of the course to previous iterations, students now were expected to complete 75% of the assignments each week in order to access the course exam and the course grade was fully based on the exam score, where 40% of the exam points corresponded to the worst passing grade, and 90% of the exam points corresponded to the best passing grade. In previous course iterations, the course grade has been based on a combination of completed programming assignments and the course exam. Moreover, in some previous course iterations, students have had multiple exams during the course, while this iteration had a single exam. Due to these differences, we discuss the differences between the grades of different course iterations only briefly.

3.2 Research questions

Studying the data collected during the experiment and the interviews, we answer the following research questions: **RQ1** How do students use the opportunity to view model solutions?; **RQ2** To what extent does students' use of model solutions correlate with course outcomes?; **RQ3** How do students and course assistants perceive the opportunity to see model solutions?

To answer RQ1, we study model solution access logs. We focus on whether students looked at the model solutions before or after completing assignments, and analyze whether student sub-populations differ in their use of model solutions. To answer RQ2, we study the correlation between model solution use and programming assignment completion as well as model solution use and exam outcomes. Furthermore, as some of the course participants drop out of the course, we also study the difference between the model solution usage of the population that stays in the course weekly as well as the population that does not stay in the course. Finally, to answer RQ3, we analyze the semi-structured interviews that were conducted with 16 course participants and teaching assistants and highlight the main themes from those interviews.

In the analysis, we consider a student having viewed a model solution before completing an assignment if the student has viewed a model solution before receiving full marks from the assignment. Similarly, we consider a student having viewed a model solution after completing an assignment if the student has viewed a model solution after receiving full marks from the assignment. The grading is done automatically by the automated assessment system. For our analysis, we count each view only once. That is, even if a student views a model solution to a particular assignment multiple times, it is still counted as a single view.

4 METHODOLOGY AND RESULTS

4.1 Descriptive statistics

In total, 342 students signed up to the course in our University course registration system. From the 342 students, 314 showed up and continued to the second week of the course. From the 314, who we count as having participated, 262 continued working on the course assignments until the end of the course, and 242 attended the course exam (77.1% of the participating students). It is possible that some of the students decided to attend an exam later, to which we do not have access at the present time.

At the beginning of the first week of the course, students were asked to provide consent for the use of their data for the analysis. From the course participants, 206 provided consent. Subsequently, the analysis focuses on the 206 students. From the 206 consenting students, 107 identified themselves as female, 92 male and 7 chose to not to disclose their gender. The median age of the students was 23, average age 25, and approximately 30% of the students had at least some programming background. From the 206 students, 167 attended the course exam (81.1% of the population who consented for the analysis).

4.2 Students' use of the opportunity to view model solutions

The course has 177 programming assignments distributed across seven weeks. The earlier weeks of the course have more assignments, but the assignments are smaller in size, while the later weeks of the course have fewer but larger assignments. This data in combination with model solution usage is summarized in Table 1.

Students use the model solutions less during the earlier weeks of the course than the latter weeks of the course. During the first week of the course with simple assignments, the average number of model solutions viewed before completing the assignment was 7.9, which corresponds to approximately 18.8% of the weekly assignments. During the last week of the course with more complex assignments, the average number of model solutions viewed before completing the assignment was 5.5, which corresponds to approximately 61.1% of the weekly assignments.

Table 1: Number of model solutions viewed before and after completing the programming assignment. Here, # corresponds to the number of programming assignments during the week and *std* corresponds to standard deviation.

		Before completion			After completion		
Week	#	median	mean	std	median	mean	std
1	42	4	7.9	9.9	1	1.9	3.3
2	33	6	10	10	1	2	3.3
3	34	11	12.6	11	1	2.3	3.9
4	28	8	9.7	8.5	0	1.4	2.9
5	17	8	7.7	6	0	0.8	2.2
6	14	9	8	4.8	0	0.7	1.7
7	9	6	5.5	2.9	0	0.3	0.8
All	177	47	56.6	46.6	5	9.1	14.6

Students viewed the model solutions significantly more before completing programming assignments than after completing programming assignments. During the first week of the course, the average number of model solutions viewed after completing the assignment was 1.9, which corresponds to less than 5% of the weekly assignments. Similarly, during the last week of the course, the average number of model solutions viewed after completing the assignment was almost zero.

When considering all the weeks combined (row "All" in Table 1), students on average viewed 56.6 model solutions before completing assignments, which corresponds to 32.0% of the total number of programming assignments in the course. The lower trend in the number of times that model solutions were viewed after completing programming assignments was visible here as well. On average, students viewed a model solution 9.1 times out of the 177 assignments, after completing the assignment, which corresponds to 5.1% of the total number of programming assignments in the course.

We also studied if the number of model solutions viewed before completing an assignment could be explained by (1) gender, (2) previous programming experience, or (3) major. Using Kolmogorov-Smirnov -test with Bonferroni correction to compare the use of model solutions before completing the assignment, we found no statistically significant difference in gender ($p = 0.07$). There was a statistically significant difference in the use of model solutions when comparing students with at least some programming experience to those with no programming experience ($p = 0.012$); students with some programming experience viewed less model solutions. When comparing students' major, we found a statistically significant difference in the use of model solutions between students who major in computer science and students who major in other subjects ($p < 0.001$). Computer science majors view fewer model solutions than students who major in other subjects – students in other subjects viewed 42% more model solutions before completing the assignments.

4.3 Model solution use and course outcomes

Next, we studied the model solution usage and course outcomes. We first study the connection between model solution usage and assignments completed in the course, and then model solution usage and dropping out of the course. Finally, we study model solution use and exam outcomes.

First, using Spearman's rank correlation with Bonferroni correction, we found no statistically significant correlation between the number of completed assignments and the number of model solution views before completing programming assignments ($r = 0.08, p = 0.25$). There was a moderate statistically significant correlation between viewing the model solutions after completing programming assignments and the number of completed programming assignments ($r = 0.41, p < 0.0001$).

Second, we analyzed whether dropping out of the course could be explained to some extent by viewing model solutions. For each course week from 1 to 6, we divided the students into two populations – those who completed assignments in the subsequent week and those who did not – both populations had completed a comparable number of course assignments in the current week. Then, for the populations each week, we studied their use of model solutions to determine if the model solution usage could explain students dropping out. The comparison of the populations was done using the Kolmogorov-Smirnov test with Bonferroni correction. No statistically significant difference in terms of model solution usage between the populations that dropped out of the course and the population that stayed in the course was observed ($week1 - week6, p > 0.05$). Here, however, the average number of students dropping out each week was 6, which biases the comparison due to the small number.

Finally, we analyzed students' use of model solutions and exam outcomes. Focusing on the 167 students in the exam with consent, we studied the correlation between the number of viewed model solutions before and after completing the assignments and the exam outcomes. Using Spearman's rank correlation with Bonferroni correction, we identified a strong negative correlation between the number of viewed model solutions before completing assignments and the course exam ($r = -0.71, p < 0.0001$). There was no correlation between the number of viewed model solutions after completing assignments and the course exam ($r = 0.01, p = 0.93$). Figure 1 illustrates students' exam points and model solution views before completing assignments.

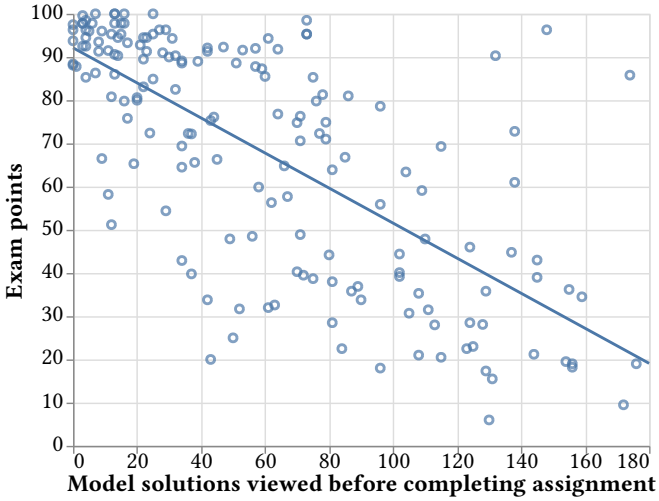

Figure 1: The number of model solution views before completing an assignment plotted with exam points. Note that a student may have viewed a model solution for a particular assignment multiple times, but our analysis calculates that as a single view event.

4.4 Students' and teaching assistants' views on the availability of model solutions

We then analyzed the 16 semi-structured interviews with the course participants and teaching assistants to study how the experiment was perceived. Overall, feelings were mixed, and the majority of the interviewed saw both benefits and downsides to the experiment. Here, we first outline the positive observations regarding the experiment and then discuss the negative observations.

4.4.1 Positive observations. Both students and teaching assistants felt that model solutions were highly beneficial if not abused – the importance of self-regulation and meta-cognition rose multiple times as themes in the interviews, albeit students did not use the actual terms. Some also pointed out that, when compared to other courses, they thought more about their learning process due to the availability of model solutions.

Several mentioned having been stuck for a while at a rather simple problem, for example due to a typo, which was then easily noticed and fixed when comparing their solution with the model solution. Issues with, for example, the expected formatting of the output could take plenty of time to fix without being able to use the model solutions. One student mentioned an assignment, where the program was expected to output `"The sum is: " + number` and the student had written `"The sum is " + number`. The student pointed out that figuring out the mistake was easy with the model solution, but that they had spent plenty of time trying to find bugs in the program logic.

A common theme was also the benefit of studying model solutions. Some pointed out that studying model solutions, either before or after completing the assignment, provided an opportunity for reflection. Moreover, model solutions were seen as a good source for learning and a good source for new approaches to solving a problem, especially if the students' own solution was different than the model solution.

There were also students who had attended the course previously or who had programmed previously. Some mentioned that the use of model solutions was a simple way to jump forward in the course – a student could copy the solution to "a boring problem" and focus on assignments that they felt were not trivial to them. This behavior, however, also requires insight into the students' own level of competence. Finally, some also pointed out that the availability of model solutions helped them stay on the course. There were those who had taken too many courses, and model solutions made it possible to handle the worst time conflicts. These students often, however, noted that they would then have to revisit the problems and do the actual studying afterward.

4.4.2 Negative observations. First, many considered the experiment very unorthodox and thought about the model solutions as "an easy way for others to cheat" and "an easy way to lie to yourself". Some simply pointed out that they thought that the whole experiment was "stupid", often pointing out that the approach was completely different to what they were used to. Some pointed out that the availability of model solutions opened a highway on which students could avoid learning but still proceed in the course, also obfuscating what was supposed to be learned.

Some noted that the model solutions themselves were not always helpful. They were highly helpful when fixing simple issues such as typos and formatting output, but "when the problem was bigger, the model solution was often of no use". This was especially evident in cases, where the student's own – not yet complete – solution was very different to the model solution. Here, students could become even more confused when viewing the model solution.

One unexpected observation came from the teaching assistants. When compared to the previous course iterations, the attendance in the walk-in labs had reduced significantly. Where previously the teaching assistants often had many students to help, the experiment made the labs feel redundant at times. Many of the students had replaced the walk-in labs and the teaching assistants with the model solutions, potentially also removing the possibility to discuss their solutions with others. Even with this observation, some teaching assistants pointed out that the availability of model solutions is not a bad thing by itself, but there should be some way to limit the access to them – for example, by limiting the number of times that model solutions could be accessed each week.

5 DISCUSSION

5.1 Model solutions were popular

Overall, the use of model solutions was popular and almost every student viewed a model solution at least once. On average, students viewed the model solution of over 50 programming assignments out of 177 before completing the programming assignment. Students were more likely to view the model solution of an assignment in the later weeks of the course than in the earlier weeks. It is possible that this was caused by the increasing complexity of the assignments.

We also observed that students were more likely to view model solutions before completing the programming assignment than after completing the programming assignment. Model solutions were viewed after completing the assignment significantly fewer times than before completing the assignment. One should note, however, that we did not study whether the students who viewed the model solutions before completing the assignment and the students who viewed the model solutions after completing the assignment were the same. That is, for some of the students who have viewed the model solution before completing the assignment, viewing the assignment afterward can be redundant. Similarly, there may be some students who complete the assignment but do not submit it for grading – instead, they may compare their completed but not submitted solution with the model solution before actually submitting the work; our current analysis which is based on the submission and model solution view timestamps cannot determine whether such behavior existed.

When looking at the differences in the use of model solutions between gender, previous programming experience, and major, we did not identify any significant difference in the use of model solutions that could be explained by gender. Our analysis, however, showed that there was a difference in the use of model solutions between students who have at least some previous programming experience and those who have no previous programming experience. Similar difference was identified in students' majors – students who major in Computer Science are less likely to view model solutions than students majoring in other subjects. Computer Science majors viewed considerably fewer model solutions than others, which may suggest that the students who major in the subject are more conscious of their learning, possibly due to knowing that the programming course is a prerequisite to many subsequent computer science courses, but it is possible also that they have more previous exposure to programming. In our case, the course had also some students for whom computer science is a mandatory minor – it is possible that some of these students have sought to put in the minimum effort to the course through the use of model solutions.

5.2 Model solutions and course outcomes

When analyzing the correlations between the number of assignments the students completed and the number of times the students viewed model solutions before assignments, no statistically significant correlation was identified. However, there was a statistically significant difference between viewing model solutions after the assignment and the number of completed assignments. This could be due to more meticulous students both viewing more model solutions after the assignment and completing more assignments. Interestingly, viewing model solutions after the assignment was, however, not correlated with exam scores.

When analyzing course exam outcomes and model solution usage, we observed a strong negative correlation between model solution usage before completing the assignment and overall exam points. This indicates that the more model solutions students used, the worse their exam outcome was. When looking at the Figure 1, we observe that there are also students who do well in the course even if they use large amounts of model solutions and that there are students who do poorly in the course even if they do not use many model solutions. It is possible that the negative correlation is caused e.g. by worse performing students requiring more help – and thus consulting the model solutions more. In any case, we suggest that others attempting a similar experiment should limit the number of times that model solutions can be viewed.

5.3 Model solutions and our community of learners

In our courses that are organized using the traditional model, the walk-in labs have been quite popular with students. Through the use of the walk-in labs, we have sought to build a community of learners – a particular type of community of practice [15, 27] where new students come to the campus as legitimate peripheral participants and work on the course assignments in collaboration with others. Having this type of community has also been important as a pool of potential teaching assistants, and typically active students have been recruited to be TAs on subsequent courses.

In the current experiment, where students were able to look at the model solutions whenever they wished, the use of the walk-in labs diminished noticeably. This made the teaching assistants feel their work as somewhat unnecessary and may have also influenced whether students come to the campus. It is also possible that some of the activity that has traditionally been conducted in a physical classroom has moved to online environments – we have observed the formation of online communities of learners in the course. The change in the number of walk-in labs attendants has, regardless, been so sudden that it is likely that the experiment was the cause.

5.4 A comparison with previous course iterations

As noted in 3.1, the course organization changed due to the experiment with making model solutions visible for all. Students no longer were given points towards their grade from completing course assignments and the students were expected to complete at least 75% of the weekly assignments to be allowed to attend the exam. Students have previously had more than one exam during the course, while in our experiment there was only a single exam at the end of the course. Also, due to students having just a single exam at the end of the course, the exam was more challenging as it covered more content.

When comparing the proportion of students who started the course and attended the exam, there is no noticeable difference between the course iterations. In the experiment reported in this article, 77.1% of the students participated in the course exam, while in the course version from last year, 77.9% of the students participated in the last exam. The students from the previous year fared better in their exam, but as the whole exam was different and somewhat easier than the current exam, a deeper analysis is not meaningful. The only somewhat comparable question in the (final) exam from last year and the (only) exam in our current experiment was a variant of the Rainfall problem [21]. In our experiment, students scored on average 88.1% in the problem, while the students in the previous year scored on average 91.8% in the problem.

Over the years, we have collected a set of "unofficial" solutions to the programming assignments offered in our course. These solutions come from a wide variety of sources ranging from Github and Pastebin-like services to solutions that students have admitted to having plagiarized from elsewhere. While in our typical courses a handful of students are found to have used these unofficial sources, in the current experiment we did not identify the use of these sources or plagiarism in the assignments. At the same time, quite a few of our students did use our official model solutions as a part of their work. That is, it seems that our experiment moved the source of the undesirable behavior to a controlled environment.

5.5 Limitations of work

There are limitations in this work, which we outline next. First, as we did not conduct a randomized controlled trial with the experiment, all of the course participants had the same material and activities. Thus, it is not possible to say which changes to the course affected the outcomes the most. We cannot, for example, say to what extent the model solutions, the different grading criteria, and the increased mandatory amount of weekly assignments contributed to students' behavior. As we recruited the interviewed students and teaching assistants from the course labs, there is a possible selection bias. It is possible that the interviewed students were more active students in the course. We do not, however, have their grade information and cannot explore this further. Finally, there is a population bias as the study was conducted in Finland, a country with a rather homogeneous population, which also tends to rank highly when examining trust between individuals and trust of authorities, which may influence the use of model solutions – it is not certain whether the results obtained here generalize to other contexts.

6 CONCLUSIONS AND FUTURE WORK

In this article, we described an experiment where students could view model solutions to programming assignments whenever they wished, regardless of whether they had completed the assignment. Our motivation was to give students the possibility to avoid unproductive time spent stuck on an assignment and to possibly facilitate learning. Additionally, having the option could give students more ownership of their learning, and being able to see an "official" solution can act as a worked example and reduce unwanted plagiarism.

Our results suggest that if model solutions are made available, students will use them. In our case, on average, students viewed the model solution of almost 30% of the course assignments before completing the assignment. There was no significant difference in the way how students with and without previous programming experience used the model solutions, nor was there any significant difference in gender. We did observe, however, that the students who are majoring in computer science use the model solutions less than others, which may suggest that students who know that they will need the knowledge in the future may control their learning better. Overall, our experiment also suggests that the more model solutions the students use before completing assignments the worse they will perform in the course exam on average.

To summarize, our answers to the research questions are as follows. **RQ1** How do students use the opportunity to view model solutions? **Answer:** *Many refer to the model solutions when completing programming assignments. At the same time, very few look at the model solutions after completing assignments.* **RQ2** To what extent does students' use of model solutions correlate with course outcomes? **Answer:** *Viewing model solutions before completing assignments correlates negatively with course outcomes, while viewing model solutions after completing assignments seems to not be connected with exam outcomes.* **RQ3** How do students and course assistants perceive the opportunity to see model solutions? **Answer:** *Many see benefit in making model solutions available, however, many also note that students can abuse model solutions.*

In our ongoing work, we have chosen not to continue with the experiment of giving unrestricted access to model solutions. The main reason for this decision was the observation that students who used a lot of model solutions performed worse in the course exam. We do still provide students access to model solutions, but instead of being able to use them without any constraint, students have access to a limited number of model solutions during the course – this may prevent students from abusing the availability of the model solutions as the limited number of model solutions can be seen as a scarce resource that students may wish to retain [16].

REFERENCES

[1] Cheryl L Aasheim, Paige S Rutner, Lixin Li, and Susan R Williams. 2012. Plagiarism and programming: A survey of student attitudes. *Journal of information systems education* 23, 3 (2012), 297.

[2] Amjad Altadmri and Neil CC Brown. 2015. 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 522–527.

[3] John R. Anderson and Edward Skwarecki. 1986. The automated tutoring of introductory computer programming. *Commun. ACM* 29, 9 (1986), 842–849.

[4] Ryan S Baker. 2016. Stupid tutoring systems, intelligent humans. *International Journal of Artificial Intelligence in Education* 26, 2 (2016), 600–614.

[5] Peter Brusilovsky and Gerhard Weber. 1996. Collaborative example selection in an intelligent example-based programming environment. In *Proceedings of the 1996 international conference on Learning sciences*. International Society of the Learning Sciences, 357–362.

[6] Allan Collins, John Seely Brown, and Ann Holum. 1991. Cognitive apprenticeship: Making thinking visible. *American educator* 15, 3 (1991), 6–11.

[7] Allan Collins, John Seely Brown, and Susan E Newman. 1989. Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics. *Knowing, learning, and instruction: Essays in honor of Robert Glaser* 18 (1989), 32–42.

[8] Vanessa P Dennen and Kerry J Burner. 2008. The cognitive apprenticeship model in educational practice. *Handbook of research on educational communications and technology* 3 (2008), 425–439.

[9] Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2012. All Syntax Errors Are Not Equal. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '12)*. ACM, New York, NY, USA, 75–80. https://doi.org/10.1145/2325296.2325318

[10] Walter Doyle. 1983. Academic work. *Review of educational research* 53, 2 (1983), 159–199.

[11] Jeremiah M Faries and Brian J Reiser. 1988. *Access and use of previous solutions in a problem solving situation*. Technical Report. Princeton University Cognitive Science Laboratory.

[12] Katherine Fu, Jonathan Cagan, and Kenneth Kotovsky. 2010. Design team convergence: the influence of example solution quality. *Journal of Mechanical Design* 132, 11 (2010), 111005.

[13] Mike Joy and Michael Luck. 1999. Plagiarism in programming assignments. *IEEE Transactions on education* 42, 2 (1999), 129–133.

[14] Slava Kalyuga. 2009. The expertise reversal effect. In *Managing Cognitive Load in Adaptive Multimedia Learning*. IGI Global, 58–80.

[15] Jean Lave and Etienne Wenger. 1991. *Situated learning: Legitimate peripheral participation*. Cambridge university press.

[16] Michael Lynn. 1991. Scarcity effects on value: A quantitative review of the commodity theory literature. *Psychology & Marketing* 8, 1 (1991), 43–57.

[17] Ference Marton and Roger Säljö. 1976. On qualitative differences in learning: I—Outcome and process. *British journal of educational psychology* 46, 1 (1976), 4–11.

[18] Paul R Pintrich. 2000. The role of goal orientation in self-regulated learning. In *Handbook of self-regulation*. Elsevier, 451–502.

[19] Kelly Rivers and Kenneth R. Koedinger. 2017. Data-Driven Hint Generation in Vast Solution Spaces: a Self-Improving Python Programming Tutor. *International Journal of Artificial Intelligence in Education* 27, 1 (01 Mar 2017), 37–64. https://doi.org/10.1007/s40593-015-0070-z

[20] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.

[21] Otto Seppälä, Petri Ihantola, Essi Isohanni, Juha Sorva, and Arto Vihavainen. 2015. Do We Know How Difficult the Rainfall Problem is?. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research (Koli Calling '15)*. ACM, New York, NY, USA, 87–96. https://doi.org/10.1145/2828959.2828963

[22] Judy Sheard, Martin Dick, Selby Markham, Ian Macdonald, and Meaghan Walsh. 2002. Cheating and Plagiarism: Perceptions and Practices of First Year IT Students. In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '02)*. ACM, New York, NY, USA, 183–187. https://doi.org/10.1145/544414.544468

[23] John Sweller and Graham A Cooper. 1985. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction* 2, 1 (1985), 59–89.

[24] J Gregory Trafton and Brian J Reiser. 1993. Studying examples and solving problems: Contributions to skill acquisition. In *Proceedings of the 15th conference of the Cognitive Science Society*. Citeseer, 1017–1022.

[25] Kurt VanLehn. 1998. Analogy events: How examples are used during problem solving. *Cognitive Science* 22, 3 (1998), 347–388.

[26] Arto Vihavainen, Matti Paksula, and Matti Luukkainen. 2011. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 93–98.

[27] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Jaakko Kurhila. 2013. Massive increase in eager TAs: Experiences from extreme apprenticeship-based CS1. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. ACM, 123–128.

[28] Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Martin Pärtel. 2013. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. ACM, 117–122.